Neevia
docuPrinter SDK

user's manual
version 6.7

License Agreement

ELECTRONIC END USER LICENSE AGREEMENT

For One (1) Computer

This is an End User License Agreement. This is a contract. If you install this software, you must abide by the terms of this agreement. This license is applicable to all software products sold by Neevia Technology. The term software includes upgrades, modified versions or updates. This software is licensed and not sold. Only a personal, non-transferable and nonexclusive right to use the Neevia products is granted to the end user.

The following are definitions that should be noted by the user:

a. COMPUTER

This is a single computer owned, rented or leased by a single individual or entity on which one or more applications load and execute software in the memory space of that computer. Software is installed on a server for one or more users. All servers must be licensed to utilize Neevia software.

THIS IS A CONTRACT BETWEEN YOU AND NEEVIA TECHNOLOGY. YOU SHOULD CAREFULLY READ THIS LICENSING AGREEMENT AND MUST ACCEPT ALL THE TERMS AND CONDITIONS BEFORE INSTALLING THIS NEEVIA SOFTWARE. BY INSTALLING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS AND CONDITIONS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, DO NOT INSTALL THE SOFTWARE, AND DO NOT USE THE SOFTWARE. IF YOU VIOLATE THIS AGREEMENT, YOU WILL BE SUBJECT TO LEGAL ACTION BY NEEVIA TECHNOLOGY.

Subject to the payment of applicable license fees, Neevia Technology grants you a nonexclusive right to use its accompanying Neevia software product and related documents (the Software) in the terms and conditions provided as follow:

LICENSE

Until such time as Neevia Technology has issued a valid serial number to you, you may only use this software for a 30-day trial period. You agree to remove any copies of the software after the expiration of the trial period. No license is issued to you until you are issued a valid serial number.

You cannot use a license for the software concurrently on different computers. You may install and use the Software in a single location on a hard disk or other storage device of one computer only.

(a) Personal Use:

The primary user of each computer on which the Software is installed or used may also install the Software on one home or portable computer. However another person may not use the Software on a secondary computer at the same time the Software on the primary computer is being used.

(b) Server or Network Use:

You may keep one copy of the Software on a single file server only for the purposes of downloading and installing the Software onto a hard disk of up to the Permitted Number of Computers that are on the same network as the file server. No other network use is permitted.

(c) Operating system or Language versions:

If you receive two or more copies of the Software with different operating systems or language versions, the total aggregate number of computers on which all versions of the Software are used may not exceed the Permitted Number of Computers. You may not rent, lease, sublicense, lend or transfer versions or copies of the Software you do not use, or Software contained on any unused media.

(d) Archiving:

You may make one copy of the Software solely for archival purposes. If the Software is an upgrade, you may use the Software only in conjunction with upgraded product. If you receive your first copy of the Software electronically, and a second copy on media afterward, the second copy can be used for archival purposes only.

For all Neevia Technology products, you agree that you will only use our software on a server and all applications that will access the server will reside on the server and you will not permit remote access to the software except through your application residing on the server. You agree to surrender your license(s) if you violate this agreement. If you violate this agreement, you will not receive a refund upon termination of this license. You agree not to utilize our software to violate the copyright of any third parties. If you do violate the copyright of a third party utilizing our software, you agree to hold Neevia Technology harmless and will indemnify Neevia Technology for any such activity even if the violation is unintentional.

COPYRIGHT

The Software is owned by Neevia Technology and/or its suppliers, and is protected by the copyright and trademark laws of the United States and related applicable laws. You may not copy the Software except as set forth in the "License" section. Any copies that you are permitted to make pursuant to this Agreement must contain the same copyright and other proprietary notices that appear on or in the Software.

You may not rent, lease, sub-license, transfer, or sell the Software. You may not modify, translate, reverse engineer, decompile, disassemble, or create derivative works based on the Software, except to the extent applicable law expressly prohibits such foregoing restriction. You may use the trademarks to identify the Software owner's name, or to identify printed output produced by the Software. Such use of any trademark does not give you any rights of ownership in that trademark.

NO WARRANTY LICENSED SOFTWARE (S) - "AS IS"

The Software is provided AS IS. NEEVIA TECHNOLOGY AND ITS SUPPLIERS MAKE NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE MERCHANTABILITY, QUALITY, NONINFRINGEMENT OF THIRD PARTY RIGHTS, FITNESS FOR A PARTICULAR PURPOSE, AND THOSE ARISING BY STATUTE OR OTHERWISE IN LAW OR FROM A COURSE OF DEALING OR USAGE OF TRADE. THE ENTIRE RISK AS TO THE QUALITY, RESULTS BY USING THE SOFTWARE, AND PERFORMANCE OF THE SOFTWARE IS WITH THE END USER. Some states or jurisdictions do not allow the exclusion or limitation of incidental, consequential or special damages, or the exclusion of implied warranties or limitations on how long an implied warranty may last, so the above limitations may not apply to you or your company.

LIMITATION OF REMEDIES AND LIABILITY

NEEVIA TECHNOLOGY OR ITS SUPPLIERS OR RESELLERS SHALL NOT UNDER ANY CIRCUMSTANCE BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST SAVINGS, OR FOR ANY CLAIM BY A THIRD PARTY, ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE, EVEN IF NEEVIA TECHNOLOLGY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

GENERAL

This Agreement shall be construed, interpreted, and governed by the laws of the State of Florida, excluding the application of its conflicts of law rules. The United Nations Convention on Contracts of the International Sale of Goods, will not govern this Agreement. If any part of this Agreement is found void and unenforceable, it will not affect the validity of the rest of the Agreement, which shall remain valid and enforceable according to its terms.

If you need to redistribute this product with your own software products, you need to contact Neevia and negotiate a separate licensing and royalty agreement.

You may not ship, transfer, or export the Software into any country or used in any manner prohibited by any export laws, restrictions or regulations.

UPGRADES

You must be properly licensed to install upgrades to Neevia Software products. Neevia upgrades replace and or supplement the previous product that formed the basis for your eligibility to for the upgrade. You may use the upgrade only in accordance with the terms of this Agreement. Upgrades may not be separated and used on separate computers.

GOVERNEMENT USERS

For United States government users, the Software and associated Documentation are deemed to be "commercial computer software" and "commercial computer documentation", respectively pursuant to DFAR 227.7202 and FAR 12.212(b) as applicable.

ENTIRE AGREEMENT

You acknowledge that you have read this Agreement, understand it and agree to be bounded by its terms and conditions. It is the complete and exclusive statement of the Agreement between us, which supersedes any proposal or prior agreement, oral or written, and other communication between us relating to the subject matter of this Agreement.

CONTACT INFORMATION

NEEVIA TECHNOLOGY
Tel: (954) 893.9343
Email: **info@neevia.com**
Web: **http://neevia.com**

Table of Contents

Title page **1**
License Agreement **2**
Table of Contents **5**
Introduction **9**
System Requirements 9
Output Formats 9
Installing and Uninstalling Neevia docuPrinter SDK **10**
Neevia docuPrinter SDK COM object **12**
Neevia docuPrinter SDK .NET assembly **12**
Neevia docuPrinter SDK Type Library (TLB file) **12**
Properties **13**
DocumentOutputFolder 13
DocumentOutputName 13
DocumentOutputFormat 13
DocumentResolution 13
PDF Specific Properties **14**
DocumentTitle 14
DocumentAuthor 14
DocumentSubject 14
DocumentKeywords 14
OptimizePDFfor 14
PDFLinearized 15
PDFCompatibilityLevel 15
PDFAutoRotatePage 16
PDFEmbedAllFonts 16
PDFProcessColorModel 16
PDFCompressPages 16
PDFSubsetFonts 16
PDFFontsMaxSubset 17
ConvertCMYKImagesToRGB 17
CompressColorImages 17
CompressGrayImages 17
CompressMonoImages 17
ColorCompressMethod 18
GrayCompressMethod 18
MonoCompressMethod 18
ColorImageResolution 18
GrayImageResolution 18
MonoImageResolution 19
DownsampleColorImages 19
DownsampleGrayImages 19
DownsampleMonoImages 19
MaxInlineImageSize 19
ColorImageDownsampleType 20
GrayImageDownsampleType 20
MonoImageDownsampleType 20
PDF/A Specific Properties 21

OutputIntent 21
PDF Encryption Properties 22
PDFEncryption 22
PDFEncryptionType 22
PDFUserPassword 22
PDFOwnerPassword 22
PDFNoCopyPermission 23
PDFNoPrintPermission 23
PDFNoChangePermission 23
PDFNoAddPermission 23
PDFContentAccess 23
PDFAllowExtraction 24
PDFChangesAllowed 24
PDFPrintingPermissions 24
PDF Viewer specific properties 25
HideToolBar 25
HideMenuBar 25
HideMainWindow 25
PDFOpenView 25
PDFOpenAtPage 25
PDFOpenMagnification 26
PDFPageLayout 26
Watermark and Stationery Properties 27
StampFontName 27
StampFontColor 27
StampFontSize 27
StampMessage 27
StampDrawMode 28
PlaceStampOnPages 28
Watermark 28
Stamp_X 28
Stamp_Y 29
StampUnits 29
StampOpacity 29
StampRotate 29
StationeryFile 29
PlaceStationeryOnPages 29
StationeryAsWatermark 30
Stationery_X 30
Stationery_Y 30
StationeryUnits 30
StationeryOpacity 30
StationeryRotate 30
Image specific properties 31
ImageType 31
JPEGImageQuality 32
MultipageTiff 32
TiffFillOrder 32
TextAplhaBits 32
GraphicsAplhaBits 33

UseCIEColor 33
Interpolate 33
UseWTS 33
FileMask 34
User Interface specific properties 35
HideSaveAsWindow 35
HideWatermarkButton 35
HideConfigureButton 35
HideEmailCheckBox 35
HideViewCheckBox 35
HidePDFdesktopCheckBox 35
DefaultAction 36
ViewDocumentAfterConversion 36
EmailDocumentAfterConversion 36
OpenInPDFdesktopAfterConversion 36
Methods 37
NewGUID 37
GetDefaultPrinter 37
SetDefaultPrinter 37
BackupSettings 37
RestoreSettings 37
ApplySettings 37
Create 38
TrueTypeFontDownloadOption 38
FileDelete 38
FileInUse 38
FileExists 38
FileCopy 38
docuPrinter Word macro 39
CHBookmarks 39
CInternetLink 39
CCrosRef 39
CCrosDoc 39
LinkFootEnd 40
CWordTextBox 40
CDocInfo 40
LinkType 40
LinkHighlight 40
LinkColor 41
LinkStyle 41
CBookNameDest 41
BookmarkDepth 41
BookmarkMagn 41
CFormFields 42
RenameFormFields 42
HideTextInputBorder 42
HideCheckBoxBorder 42
HideDropDownBorder 42
CComNotes 42
ConvertDocument 42

docuPrinter Excel macro **43**
CHBookmarks 43
CInternetLink 43
CDocInfo 43
LinkType 43
LinkHighlight 44
LinkColor 44
LinkStyle 44
CBookNameDest 45
BookmarkDepth 45
BookmarkMagn 45
ConvertDocument 45
docuPrinter PowerPoint macro **46**
CTransition 46
CHBookmarks 46
CInternetLink 46
CDocInfo 46
LinkType 47
LinkHighlight 47
LinkColor 47
LinkStyle 47
BookmarkDepth 48
BookmarkMagn 48
BlackAndWhitePrinting 48
ConvertDocument 48
Examples **49**
Example 1. Create a PDF file from Visual Basic 49
Example 2. Convert a MS Word document into PDF from Visual Basic 50
Example 3. Convert a MS Word document into PDF from Visual Basic (using the docuPrinter MS Word Macro) 51
Example 4. Convert a MS Excel document into PDF from Visual Basic 52
Example 5. Convert a MS PowerPoint document into PDF from Visual Basic 53
Example 6. Convert a PowerPoint document into PDF from Visual Basic 54
Example 7. Convert a MS Access report into PDF from Visual Basic 55
Example 8. Convert an URL / HTML into PDF from Visual Basic 56

Introduction

docuPrinter SDK is a software development tool that can be used by developers and programmers to control **docuPrinter LT, Pro or TSE** and programmatically create PDF or Image files from their own applications.

docuPrinter SDK works on every Windows operating system from Windows 2000 to Windows Server 2008 and because it includes both C/C++ libraries and ActiveX controls, the functionality of the product can be accessed from most programming languages like C, C++, Visual Basic, Delphi, MS FoxPro, and MS Access. docuPrinter SDK is also .NET compatible meaning that VB.NET, C#, and J# programmers can also take full advantage of the product.

System Requirements

Supported Operation Systems

The operating systems listed below (both 32 and 64 bit) have been tested with docuPrinter SDK and are officially supported:

Windows XP, 2003, Vista, 7, 2008, 2012, 8, 10.

Recommended hardware

- Standard PC; 500MHz or faster compatible x86 processor;
- **RAM:** 512MB RAM minimum;
- **Hard-disk:** 15MB free space recommended;

Output Formats

If you have docuPrinter LT installed on the computer, docuPrinter SDK only supports **PDF, PDF/A** and **PostScript** as output formats. If you have docuPrinter Pro or TSE installed, docuPrinter SDK supports the following output formats:

- **PDF** (Portable Document Format) v1.2, v1.3, v1.4, v1.5, v1.6, v1.7;
- **PDF/A-1b**
- **PostScript** level 1, 1.5, 2, 3;
- **TIFF** (uncompressed, LZW, Packbits, G3, G4);
- **JPEG**;
- **PCX**;
- **PNG**;
- **BMP**;
- **EPS** level 1, 1.5, 2, 3;
- **PSD** (Adobe PhotoShop);
- HP **PCL-XL** (color and mono);
- **Text**;

Installing and Uninstalling Neevia docuPrinter SDK

Before installing and/or using this product, please make sure you carefully read the copyright notice and agree to all of its terms. If you have any questions about the licensing agreement, please feel free to call (954) 981.9252 or email sales@neevia.com.

If you are using an earlier version of Neevia docuPrinter SDK on your system, it is recommended that you uninstall it before installing Neevia docuPrinter SDK v6.5.

To install Neevia docuPrinter SDK:

If you already have docuPrinter LT, PRO or TSE installed on your computer:
download and save the http://neevia.com/prods/dpsdk.exe file to your hard disk. After downloading the file, double-click it and follow the instructions. The installation procedure automatically detects your operating system, copies the needed files to your system directory and installs Neevia docuPrinter SDK.

If you don't have docuPrinter LT, PRO or TSE installed on your computer:
download and install one of the above products then download and save the http://neevia.com/prods/dpsdk.exe file to your hard disk. After that double-click on it and follow the instructions. The installation procedure automatically detects your operating system, copies the needed files to your system directory and installs Neevia docuPrinter SDK.

Unattended installation:

To perform an unattended (silent) installation launch docuPrinter SDK installer with **/sp- /verysilent /norestart** command line switches. Here is the full list of supported switches:

/SP-

Disables the "This will install... Do you wish to continue?" prompt at the beginning of Setup.

/SILENT, /VERYSILENT

Instructs Setup to be silent or very silent. When Setup is silent the wizard and the background window are not displayed but the installation progress window is. When a setup is very silent this installation progress window is not displayed. Everything else is normal so for example error messages during installation are displayed and the startup prompt is (if you haven't disabled it with '/SP-' command line option explained above). If a restart is necessary and the **'/NORESTART'** command isn't used (see below) and Setup is silent, it will display a "Reboot now?" dialog. If it's very silent it will reboot without asking.

/NOCANCEL

Prevents the user from canceling during the installation process, by disabling the Cancel button and ignoring clicks on the close button. Useful along with /SILENT.

/NORESTART

Instructs Setup not to reboot even if it's necessary.

/DIR="x:\dirname"

Overrides the default directory name displayed on the Select Destination Directory wizard page. A fully qualified pathname must be specified.

/GROUP="folder name"

Overrides the default folder name displayed on the Select Start Menu Folder wizard page.

/user="username", **/company=**"company name", **/serial=**"serial number"

Use these switches to pass the registration info (username, company name and serial number) to the installer.

To remove Neevia docuPrinter SDK from your system:

1. Select **Settings -> Control Panel** from the **Start** menu.
2. In the Control Panel click **Add/Remove programs** and select **docuPrinter SDK** from the list.
3. Click the **Add/Remove** button to remove the program.

A confirmation prompt is displayed.

Unattended uninstall:

To perform an unattended (silent) uninstall, launch **unins000.exe** from the folder where docuPrinter SDK has been installed, with **/verysilent /norestart** command line switches. Here is the full list of supported switches:

/SILENT, /VERYSILENT

When specified, the uninstaller will not ask the user for startup confirmation or display a message stating that uninstall is complete. Shared files that are no longer in use are deleted automatically without prompting. Any critical error messages will still be shown on the screen. When '/VERYSILENT' is specified, the uninstallation progress window is not displayed. If a restart is necessary and the '/NORESTART' command isn't used (see below) and '/VERYSILENT' is specified, the uninstaller will reboot without asking.

/NORESTART

Instructs the uninstaller not to reboot even if it's necessary.

NOTE: Before calling the docuPrinter SDK COM object, make sure that you have docuPrinter LT, PRO or TSE installed!

Neevia docuPrinter SDK COM object

Class ID

docuPrinter.SDK

Example:

Visual Basic: **Set DPSDK = CreateObject("docuPrinter.SDK")**

Delphi: **DPSDK := CreateOLEObject('docuPrinter.SDK');**

ASP: **Set DPSDK = Server.CreateObject("docuPrinter.SDK")**

VC#: first add a reference in your project to docuPrinter library

docuPrinter.SDK DPSDK = new docuPrinter.SDK();

Neevia docuPrinter SDK .NET assembly

The docuPrinter SDK .NET assembly is located under **C:\program files (x86)\neevia.com\docuPrinterSDK\.Net**

Neevia docuPrinter SDK Type Library (TLB file)

The docuPrinter SDK Type Library is located under **C:\program files (x86)\neevia.com\docuPrinterSDK\TLB**

Properties

DocumentOutputFolder

Sets the output folder.

Syntax

DPSDK.DocumentOutputFolder = value

Data type: **String**

Note: If the output folder doesn't exist it will be automatically created.

DocumentOutputName

Sets the output document name.

Syntax

DPSDK.DocumentOutputName = value

Data type: **String**

Note: You can use the following variables **%title%, %date%, %guid%, %time%** in the document name.
%title% - will insert the document title as sent from the printing application;
%date% - will insert the current date;
%guid% - will insert a unique identifier;
%time% - will insert the current time;

DocumentOutputFormat

Sets the output format.
Possible values: **"PDF", "PDFA", PDFA2 "PS", "EPS", "TXT", "JPG", "TIF", "PNG", "PCX", "BMP", "PSD", "PXL".**

Syntax

DPSDK.DocumentOutputFormat = value

Data type: **String**

Note: docuPrinter LT only supports "PDF", "PDFA" "PDFA2" and "PS" output formats.
Note: when PDFA is specified the output will be PDF/A-1b. To obtain a PDF/A-2b file you'll have to use PDFA2.

DocumentResolution

Sets the output resolution (in dpi).

Syntax

DPSDK.DocumentResolution = value

Data type: String

Note: to specify horizontal and vertical resolutions insert "x" between them, for ex: DPSDK.DocumentResolution="300x300".

PDF Specific Properties

DocumentTitle

Sets the title field in the output document.

Syntax

DPSDK.DocumentTitle = value

Data type: **String**

Note: Will have effect only if DocumentOutputFormat=**"PDF".**

DocumentAuthor

Sets the author field in the output document.

Syntax

DPSDK.DocumentAuthor = value

Data type: **String**

Note: Will have effect only if DocumentOutputFormat=**"PDF".**

DocumentSubject

Sets the subject field in the output document.

Syntax

DPSDK.DocumentSubject = value

Data type: **String**

Note: Will have effect only if DocumentOutputFormat=**"PDF".**

DocumentKeywords

Sets the keywords field in the output document.

Syntax

DPSDK.DocumentKeywords = value

Data type: **String**

Note: Will have effect only if DocumentOutputFormat=**"PDF".**

OptimizePDFfor

Possible values: **"default", "screen", "printer", "prepress".**

Syntax

DPSDK.OptimizePDFfor = value

Data type: **String**

Note: Will have effect only if DocumentOutputFormat=**"PDF".**

For your convenience there are several sets of predefined settings for creating PDF files.

These settings are designed to balance file size with quality, depending on how the PDF file will to be used:

- **default** - is intended to be useful across a wide variety of uses, possibly at the expense of a larger output file. All color and grayscale images are downsampled at 72 dpi, monochrome images at 300 dpi; subsets of all fonts used in the file are embedded; and all information is compressed. PDF files created using this setting are compatible with Acrobat 4.0 (and later).

• **screen** - is for PDF files that will be displayed on the World Wide Web or an intranet, or that will be distributed through an e-mail system for on-screen viewing. This set of options uses compression, downsampling, and a relatively low resolution; converts all colors to RGB; maintains compatibility with Acrobat 3.0; to create a PDF file that is as small as possible.

• **printer** - is for PDF files that are intended for desktop printers, digital copiers, publishing on a CD-ROM, or to send to a client as a publishing proof. In this set of options, file size is still important, but it is not the only objective. This set of options uses compression and downsampling to keep the file size down, tags everything for color management, and prints to a medium resolution to create a reasonably accurate rendition of the original document.

• **prepress** - is for PDF files that will be printed as high-quality final output to an imagesetter or platesetter, for example. In this case, file size is not a consideration. The objective is to maintain all the information in a PDF file that a commercial printer or service bureau will need to print the document correctly. This set of options downsamples color and grayscale images at 300 dpi, monochrome images at 1200 dpi, embeds subsets of all fonts used in the file, prints to a higher resolution, and uses other settings to preserve the maximum amount of information about the original document.

PDFLinearized

Specifies whether the output PDF document should be linearized or not.
PDF linearization is a way to optimize PDF files for more efficient viewing over the web. Pages are loaded faster and the user does not have to wait until the entire PDF file is uploaded by the browser.

Possible values*:* **true, false**

Syntax

DPSDK.PDFLinearized = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and will not work if PDFEncryption=true.

PDFCompatibilityLevel

Sets the compatibility level (a.k.a. PDF version) of the output PDF document.
Possible values:

1.2 (Acrobat Reader v3-and-later compatible)
1.3 (Acrobat Reader v4-and-later compatible)
1.4 (Acrobat Reader v5-and-later compatible)
1.5 (Acrobat Reader v6-and-later compatible)
1.6 (Acrobat Reader v7-and-later compatible)
1.7 (Acrobat Reader v8-and-later compatible)

Syntax

DPSDK.PDFCompatibilityLevel = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

PDFAutoRotatePage

Sets the orientation of the output document based on text content.
Possible values:

None	(will disable the Auto-Rotate page option)
PageByPage	(will rotate each page based on the direction of the text on that page)
All	(will rotate all pages in the document based on the orientation of the majority of text)

Syntax

DPSDK.PDFAutoRotatePage = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

PDFEmbedAllFonts

Specifies whether the fonts in the output document should be embedded or not.
Possible values: **true, false**

Syntax

DPSDK.PDFEmbedAllFonts = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

PDFProcessColorModel

Sets the color model for the output document.
Possible values: **"DeviceRGB", "DeviceCMYK", "DeviceGRAY".**

Syntax

DPSDK.PDFProcessColorModel = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and will not affect images and shadings.

PDFCompressPages

Specifies whether text and line art in the output document should be compressed or not.
Possible values: **true, false**

Syntax

DPSDK.PDFCompressPages = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

PDFSubsetFonts

Indicates whether or not to include in the output document only those characters from a font that are used in the document.
Possible values: **true, false**

Syntax

DPSDK.PDFSubsetFonts = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

PDFFontsMaxSubset

Sets the SubsetFonts threshold. If the percentage of used characters (compared with total characters of the particular font) exceeds this threshold, the entire font is embedded.
Possible values: **0...100.**

Syntax

DPSDK.PDFFontsMaxSubset = value

Data type: **Long**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

ConvertCMYKImagesToRGB

Specifies whether the CMYK images in the output document should be converted to RGB or not.
Possible values: **true, false**

Syntax

DPSDK.ConvertCMYKImagesToRGB = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

CompressColorImages

Specifies whether the color images in the output document should be compressed or not.
Possible values: **true, false**

Syntax

DPSDK.CompressColorImages = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

CompressGrayImages

Specifies whether the gray images in the output document should be compressed or not.
Possible values: **true, false**

Syntax

DPSDK.CompressGrayImages = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

CompressMonoImages

Specifies whether the mono images in the output document should be compressed or not.
Possible values: **true, false**

Syntax

DPSDK.CompressMonoImages = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

ColorCompressMethod

Sets the color images compression method.
Possible values: **"Automatic", "JPEG-maximum", "JPEG-high", "JPEG-medium", "JPEG-low", "JPEG-minimum", "ZIP".**

Syntax

DPSDK.ColorCompressMethod = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

GrayCompressMethod

Sets the gray images compression method.
Possible values: **"Automatic", "JPEG-maximum", "JPEG-high", "JPEG-medium", "JPEG-low", "JPEG-minimum", "ZIP".**

Syntax

DPSDK.GrayCompressMethod = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

MonoCompressMethod

Sets the monochrome images compression method.
Possible values*:*

CCITT (Compress monochrome images using the CCITT group 4 fax compression method)
ZIP (Compress monochrome images using ZIP-compatible compression.)

Syntax

DPSDK.MonoCompressMethod = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

ColorImageResolution

Sets the resolution for color images in the output document.
Possible values: **10...2400.**

Syntax

DPSDK.ColorImageResolution = value

Data type: **Long**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and DownsampleColorImages=**true.**

GrayImageResolution

Sets the resolution for gray images in the output document.
Possible values: **10...2400.**

Syntax

DPSDK.GrayImageResolution = value

Data type: **Long**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and DownsampleGrayImages=**true.**

MonoImageResolution

Sets the resolution for monochrome images in the output document.
Possible values: **10...2400.**

Syntax

DPSDK.MonoImageResolution = value

Data type: **Long**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and DownsampleMonoImages=**true.**

DownsampleColorImages

Specifies whether color images in the output document should be downsampled or not.
Possible values: **true, false**

Syntax

DPSDK.DownsampleColorImages = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

DownsampleGrayImages

Specifies whether gray images in the output document should be downsampled or not.
Possible values: **true, false**

Syntax

DPSDK.DownsampleGrayImages = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

DownsampleMonoImages

Specifies whether monochrome images in the output document should be downsampled or not.
Possible values: **true, false**

Syntax

DPSDK.DownsampleMonoImages = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

MaxInlineImageSize

Specifies the maximum size of an inline image in bytes.
Default value: **4000**

Syntax

DPSDK.MaxInlineImageSize = value

Data type: **Integer**
For images larger than this size, docuPrinter will create a XObject instead of embedding the image into the context stream. Note that redundant inline images must be embedded each time they occur in the document, while multiple references can be made to a single XObject image. Therefore it may be advantageous to set a small or zero value if the source document is expected to contain multiple identical images, reducing the size of the generated PDF.

ColorImageDownsampleType

Sets the color images downsample algorithm.
Possible values: **"Bicubic", "Average", "Subsample".**

Syntax

DPSDK.ColorImageDownsampleType = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and DownsampleColorImages=**true.**
To resample color, grayscale, or monochrome images, docuPrinter combines pixels in a sample area to make one larger pixel.

You provide the resolution of your output device in dots per inch (dpi) and select the resample algorithm:

Average Downsample – this algorithm averages the pixels in a sample area and replaces the entire area with the average pixel color at the specified resolution.

Subsample – this algorithm chooses a pixel in the center of the sample area and replaces the entire area with that pixel at the specified resolution. Subsample significantly reduces the conversion time compared with downsampling but results in images that are less smooth and continuous.

Bicubic Downsample – this algorithm uses a weighted average to determine pixel color and usually yields better results than the simple averaging method of downsampling. Bicubic is the slowest but most precise method, resulting in the smoothest tonal gradations.

GrayImageDownsampleType

Sets the gray images downsample algorithm.
Possible values: **"Bicubic", "Average", "Subsample".**

Syntax

DPSDK.GrayImageDownsampleType = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and DownsampleGrayImages=**true.**

MonoImageDownsampleType

Sets the monochrome images downsample algorithm.
Possible values: **"Bicubic", "Average", "Subsample".**

Syntax

DPSDK.MonoImageDownsampleType = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and DownsampleMonoImages=**true.**

Resampling monochrome images can have unexpected viewing results, such as no image display. If this happens, turn off resampling and convert the file again. This problem is most likely to occur with subsample and least likely with bicubic downsample algorithm.

PDF/A Specific Properties

OutputIntent

Sets the document output intent.
Possible values: **"SRGB", "JC200103", "FOGRA27", "SWOP", "GRAY".**

Syntax

```
DPSDK.OutputIntent = value
```

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDFA".**

PDF Encryption Properties

All Neevia (and Adobe) products enforce the restrictions set by PDF Security (encryption). However, not all third party products fully support and respect these settings. Recipients using such third party products may be able to bypass some of the restrictions you have set.

PDFEncryption

Specifies whether the output PDF document should be encrypted or not.
Possible values: **true, false**

Syntax

DPSDK.PDFEncryption = value

Data type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF"**.

PDFEncryptionType

Sets the PDF encryption algorithm.
Possible values:

40	(40 bits encryption – Acrobat 3-and-later compatible)
128	(128 bits encryption – Acrobat 5-and-later compatible)
256	(256 bits encryption – Acrobat 9-and-later compatible)

Syntax

DPSDK.PDFEncryptionType = value

Data type: **Long**
Note: Will have effect only if DocumentOutputFormat=**"PDF"**.

PDFUserPassword

Sets the user password in the output document.
Users will be asked to enter this password before Acrobat Reader allows them to view the document.

Syntax

DPSDK.PDFUserPassword = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF"**.

PDFOwnerPassword

Sets the owner password in the output document.
Users will be forced to enter this password before Acrobat Reader allows them to change the user password and security permissions.

Syntax

DPSDK.PDFOwnerPassword = value

Data type: **String**
Note: Will have effect only if DocumentOutputFormat=**"PDF"**.

PDFNoCopyPermission

Specifies whether the user of the output document is allowed to copy text and graphics from the document.
Possible values: **true, false**

Syntax

DPSDK.PDFNoCopyPermission = value

Data Type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and PDFEncryptionType=**40**.

PDFNoPrintPermission

Specifies whether the user of the output document is allowed to print the document.
Possible values: **true, false**

Syntax

DPSDK.PDFNoPrintPermission = value

Data Type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and PDFEncryptionType=**40**.

PDFNoChangePermission

Specifies whether the user of the output document is allowed to change the document.
Possible values: **true, false**

Syntax

DPSDK.PDFNoChangePermission = value

Data Type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and PDFEncryptionType=**40**.

PDFNoAddPermission

Specifies whether the user of the output document is allowed to add or change comments and form fields in the document.
Possible values: **true, false**

Syntax

DPSDK.PDFNoAddPermission = value

Data Type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and PDFEncryptionType=**40**.

PDFContentAccess

Specifies whether the user of the output document is allowed to use the document contents, which is required to support the Acrobat Accessibility feature.
Possible values: **true, false**

Syntax

DPSDK.PDFContentAccess = value

Data Type: **Boolean**
Note: Will have effect only if DocumentOutputFormat=**"PDF"** and PDFEncryptionType=**128** or **256**.
Do not mix this property with PDFNoCopyPermission, PDFNoPrintPermission, PDFNoChangePermission and PDFNoAddPermission properties.

PDFAllowExtraction

Specifies whether the user of the output document is allowed to select and copy the document contents.
Possible values: **true, false**

Syntax

DPSDK.PDFAllowExtraction = value

Data Type: **Boolean**

Note: Will have effect only if DocumentOutputFormat=**"PDF"** and PDFEncryptionType=**128** or **256**
Do not mix this property with PDFNoCopyPermission, PDFNoPrintPermission, PDFNoChangePermission and PDFNoAddPermission properties.

PDFChangesAllowed

Specifies the allowed document changes.
Possible values:

- **0** (None - will prevent users from doing anything with the file, including filling in signature and form fields)
- **1** (Only Document Assembly - will let users insert, delete, and rotate pages, and create bookmarks and thumbnails)
- **2** (Only Form Field Fill-in or Signing - will let users sign and fill in forms, but not create them)
- **3** (Comment Authoring, Form Field Fill-in or Signing - will let users do everything described in the previous options, plus add comments)
- **4** (General Editing, Comment and Form Field Authoring - will let users do anything to the document except extract contents and print).

Syntax

DPSDK.PDFChangesAllowed = value

Data Type: **Integer**

Note: Will have effect only if DocumentOutput format=**"PDF"** and PDFEncryptionType=**128** or **256**
Do not mix this property with PDFNoCopyPermission, PDFNoPrintPermission, PDFNoChangePermission and PDFNoAddPermission properties.

PDFPrintingPermissions

Sets the output document printing permissions.
Possible values:

- **0** (Not Allowed – will prevent users from printing the document)
- **1** (Low Resolution – will let users print, but at a resolution that prevents from recreating the PDF file with different security settings. Printing may be slower because each page will be printed as a bitmapped image.)
- **2** (Fully Allowed – will let users print at any resolution, directing high-quality vector output to PostScript and other printers that support advanced high-quality printing features).

Syntax

DPSDK.PDFPrintingPermissions = value

Data Type: **Integer**

Note: Will have effect only if DocumentOutputFormat=**"PDF"** and PDFEncryptionType=**128** or **256**
Do not mix this property with PDFNoCopyPermission, PDFNoPrintPermission, PDFNoChangePermission and PDFNoAddPermission properties.

PDF Viewer specific properties

HideToolBar

Hide the toolbar when PDF is opened in Adobe Acrobat Reader.

Syntax

DPSDK.HideToolBar = value

Data type: **Boolean**

Note: Will have effect only if DocumentOutputFormat=**"PDF".**

HideMenuBar

Hide the menu bar when PDF is opened in Adobe Acrobat Reader.

Syntax

DPSDK.HideMenuBar = value

Data type: **Boolean**

Note: Will have effect only if DocumentOutputFormat=**"PDF".**

HideMainWindow

Hide the main window when PDF is opened in Adobe Acrobat.

Syntax

DPSDK.HideMainWindow = value

Data type: **Boolean**

Note: Will have effect only if DocumentOutputFormat=**"PDF".**

PDFOpenView

Specifies the output document initial view.
Possible values:

0 (Page only)
1 (Page and bookmarks)
2 (Page and thumbnails).
3 (Full screen)

Syntax

DPSDK.PDFOpenView = value

Data type: **Long**

Note: Will have effect only if DocumentOutputFormat=**"PDF".**

PDFOpenAtPage

Specifies the page output document should open at.

Syntax

DPSDK.PDFOpenAtPage = value

Data type: **Long**

PDFOpenMagnification

Specifies the open magnification for the output document.
Possible values:

0 (Default)
1 (Fit Window)
2 (Fit Width)
3 (Fit Height)
4 (Zoom 25%)
5 (Zoom 50%)
6 (Zoom 100%)
7 (Zoom 125%)

Syntax

DPSDK.PDFOpenMagnification = value

Data type: **Long**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

PDFPageLayout

Specifies the output document page layout.
Possible values:

0 (Page only)
1 (Single Page)
2 (Continuous)
3 (Continuous-Facing)

Syntax

DPSDK.PDFPageLayout = value

Data type: **Long**
Note: Will have effect only if DocumentOutputFormat=**"PDF".**

Watermark and Stationery Properties

StampFontName

Sets the watermark font name.

Syntax

DPSDK.StampFontName = value

Data type: **String**

StampFontColor

Sets the watermark font color.

Syntax

DPSDK.StampFontColor = value

Example

DPSDK.StampFontColor = "$0000FF"

Data type: **String**

StampFontSize

Sets the watermark font size (in points).

Syntax

DPSDK.StampFontSize = value

Data type: **Long**

StampMessage

Sets the watermark text message.

Syntax

DPSDK.StampMessage = value

Data type: **String**

Note: You can use the following variables:

%PAGE%	- current page number
%PAGES%	- total number of pages
%WEEKDAY%	- full weekday name
%WEEKDAY_SHORT%	- abbreviated weekday name
%MONTH%	- month number (1-12)
%MONTHNAME%	- full month name
%MONTHNAME_SHORT%	- abbreviated month name
%YEAR%	- year with century (YYYY)
%YEAR_SHORT%	- year without century (YY)
%DAY%	- day of month
%DAY_YEAR%	- day of the year (1 -366)
%HOUR%	- hour (01- 12)
%HOURS%	- hour (00-23)
%MINUTES%	- minutes (00-59)
%SECONDS%	- seconds (00-59)
%AMPM%	- AM, PM
%DATE%	- local date representation. Ex: 11/01/07
%TIME%	- local time representation. Ex: 1:17:10 PM

%DATETIME%	- local date and time
%AUTHOR%	- document Author
%TITLE%	- document Title
%SUBJECT%	- document Subject
%KEYWORDS%	- document Keywords

StampDrawMode

Specifies the text rendering mode. Possible values:

0 - Fill text, no stroke (default)
1 - Stroke text, no fill
2 - Fill then Stroke text
3 - Invisible

Syntax

DPSDK.StampDrawMode = value

Data type: **Long**

PlaceStampOnPages

Specifies the pages to place watermark on.

Syntax

DPSDK.PlaceStampOnPages = value

Example

DPSDK.PlaceStampOnPages = "0"

Data type: **String**

Note: Page numbers must be separated by commas. To place watermark on all pages specify 0.

Watermark

Place watermark under the original page content. Default: **false**

Syntax

DPSDK.Watermark = value

Data type: **Boolean**

Stamp_X

Sets the watermark X coordinate.

Syntax

DPSDK.Stamp_X = value

Data type: **String**

Note: Stamp coordinate depends on the page orientation. For portrait oriented pages the coordinate system starts in the left-bottom corner of the page.

Stamp_Y

Sets the watermark Y coordinate.

Syntax

DPSDK.Stamp_Y = value

Data type: **String**

Note: Stamp coordinate depends on the page orientation. For portrait oriented pages the coordinate system starts in the left-bottom corner of the page.

StampUnits

Specifies the measurement units to use for Stamp_X, Stamp_Y properties. Possible values:

1 inches
2 centimeters
3 millimeters
4 points (default)

Syntax

DPSDK.StampUnits = value

Data type: **Long**

StampOpacity

Specifies the stamp opacity. Default value: 100

Syntax

DPSDK.StampOpacity = value

Data type: **Long**

StampRotate

Specifies the stamp rotation angle. Default value: 0

Syntax

DPSDK.StampRotate = value

Data type: **Long**

StationeryFile

Specifies a PDF file to use as stationery.

Syntax

DPSDK.StationeryFile = value

Data type: **String**

PlaceStationeryOnPages

Specifies the pages to place stationery on.

Syntax

DPSDK.PlaceStationeryOnPages = value

Example

DPSDK.PlaceStationeryOnPages = "1, 4"

Data type: **String**

Note: Page numbers must be separated by commas. To place watermark on all pages specify 0.

StationeryAsWatermark

Place stationery under the original page content. Default: **false**

Syntax

DPSDK.StationeryAsWatermark = value

Data type: **Boolean**

Stationery_X

Sets the stationery X coordinate.

Syntax

DPSDK.Stationery_X = value

Data type: **String**

Note: Stamp coordinate depends on the page orientation. For portrait oriented pages the coordinate system starts in the left-bottom corner of the page.

Stationery_Y

Sets the stationery Y coordinate.

Syntax

DPSDK.Stationery_Y = value

Data type: **String**

Note: Stamp coordinate depends on the page orientation. For portrait oriented pages the coordinate system starts in the left-bottom corner of the page.

StationeryUnits

Specifies the measurement units to use for Stationery_X, Stationery_Y properties. Possible values:

1 inches
2 centimeters
3 millimeters
4 points (default)

Syntax

DPSDK.StationeryUnits = value

Data type: **Long**

StationeryOpacity

Specifies the stationery opacity. Default value: 100

Syntax

DPSDK.StationeryOpacity = value

Data type: **Long**

StationeryRotate

Specifies the stationery rotation angle. Default value: 0
Possible values: 0, 90, 180, 270

Syntax

DPSDK.StationeryRotate = value

Data type: **Long**

Image specific properties

ImageType

Specifies the output image subtype, colors and compression algorithm.
Possible values:

jpegcmyk	JPEG output format, CMYK colorspace
jpeg	JPEG output format, full color RGB (24 bits)
jpeggray	JPEG output format, grayscale
pngmono	PNG output format, monochrome
pnggray	PNG output format, grayscale
png16	PNG output format, 16 colors
png256	PNG output format, 256 colors
png16m	PNG output format, 16 million colors (full color)
pcxmono	PCX output format, monochrome
pcxgray	PCX output format, grayscale
pcx16	PCX output format, 16 colors
pcx256	PCX output format, 256 colors
pcx16m	PCX output format, 16 million colors (full color)
pcxcmyk	PCX output format, CMYK (32 bits)
bmpmono	BMP output format, monochrome
bmpgray	BMP output format, grayscale
bmp16	BMP output format, 16 colors
bmp256	BMP output format, 256 colors
bmp16m	BMP output format, 16 million colors
tiff12nc	TIFF output format, RGB output – 4 bits / channel, uncompressed
tiff24nc	TIFF output format, RGB output – 8 bits / channel, uncompressed
tiff32nc	TIFF output format, CMYK output – 8 bits / channel, uncompressed
tiff12nclzw	TIFF output format, RGB output – 4 bits / channel, LZW compressed
tiff24nclzw	TIFF output format, RGB output – 8 bits / channel, LZW compressed
tiff32nclzw	TIFF output format, CMYK output – 8 bits / channel, LZW compressed
tiffuncomp	TIFF output format, grayscale, uncompressed
tiffpack	TIFF output format, grayscale, Packbits (Macintosh RLE) compression
tiffg3	TIFF output format, grayscale, G3 fax compression with EOLs
tiffcrle	TIFF output format, grayscale, G3 fax compression with NO EOLs
Tiffg32d	TIFF output format, grayscale, G3 2D fax compression
tiffg4	TIFF output format, grayscale, G4 fax compression
tifflzw	TIFF output format, grayscale, LZW compression
psdrgb	PSD – Adobe PhotoShop output format, full color RGB
psdcmyk	PSD – Adobe PhotoShop output format, full color CMYK
pxlmono	PCL-XL output format, monochrome
pxlcolor	PCL-XL output format, full color

Syntax

DPSDK.ImageType = value

Data type: **String**

JPEGImageQuality

Sets the output document (JPEG) quality.
Possible values: 0 .. 100 (75 is recommended)

Syntax

DPSDK.JPEGImageQuality = value

Data type: **Long**

Note: Will have effect only if DocumentOutputFormat=**"JPG"** and will work with docuPrinter Pro and TSE.
JPEGImageQuality sets the output quality level according to the widely used IJG quality scale, which balances the extent of compression against the fidelity of the image when reconstituted. Lower values drop more information from the image to achieve higher compression, and therefore have lower quality when reconstituted.

MultipageTiff

Specifies whether docuPrinter should create multipage tiff files.
Possible values: **true, false**

Syntax

DPSDK.MultipageTiff = value

Data type: **Boolean**

Note: Will have effect only if DocumentOutputFormat=**"TIF"** and will work with docuPrinter Pro and TSE.
If you set MultipageTiff=false docuPrinter will create a single-page tiff file for each page of the original document.

TiffFillOrder

Sets the fill order for the TIFF output format.
Possible values: "**lsb2msb**", "**msb2lsb**"

Syntax

DPSDK.TiffFillOrder = value

Data type: **String**

Note: Will have effect only if DocumentOutputFormat=**"TIF"** and will work with docuPrinter Pro and TSE.

TextAplhaBits

This option controls the use of subsample antialiasing for text content. The subsampling box size should be 4 bits for optimum output, but smaller values can be used for faster rendering.
Possible Values: **0, 1, 2, 4**

Syntax

DPSDK.TextAplhaBits = value

Data type: **Integer**

Note: Will work with docuPrinter Pro and TSE.

GraphicsAplhaBits

This option controls the use of subsample antialiasing for graphics content. The subsampling box size should be 4 bits for optimum output, but smaller values can be used for faster rendering.
Possible Values: **0, 1, 2, 4**

Syntax

DPSDK.GraphicsAplhaBits = value

Data type: **Integer**
Note: Will work with docuPrinter Pro and TSE.
Note that because of the way antialiasing blends the edges of shapes into the background when they are drawn some files that rely on joining separate filled polygons together to cover an area may not render as expected with Graphics antialiasing at 2 or 4 bits. If you encounter strange lines within solid areas, try rendering that file again with Graphic antialiasing at 1 bit.

UseCIEColor

Specifies whether docuPrinter should remap the device-dependent color values through a CIE color space. This can improve the conversion of CMYK documents to RGB.
Possible Values: **true, false**

Syntax

DPSDK.UseCIEColor = value

Data type: **Boolean**
Note: Will work with docuPrinter Pro and TSE.

Interpolate

Specifies whether docuPrinter should use image interpolation. Enabling image interpolation will result in higher quality for scaled images at the expense of speed.
Possible Values: **true, false**

Syntax

DPSDK.Interpolate = value

Data type: **Boolean**
Note: Will work with docuPrinter Pro and TSE.

UseWTS

Specifies whether Well Tempered Screening algorithm should be used for halftoning.
Possible Values: **true, false**

Syntax

DPSDK.UseWTS = value

Data type: **Boolean**
Note: Will work with docuPrinter Pro and TSE.
If true, then the Well Tempered Screening algorithm is used for halftoning. Otherwise, a rational tangent algorithm is chosen, which will typically result in significant differences between the screen angle and ruling requested, and actually rendered. Currently, the performance of WTS is reasonably good when rendering to a full page buffer, but not optimized for banded mode.

FileMask

When printing documents that have more than 1 page, docuPrinter will produce an image file for each page of the original document. The FileMask allows you to control how the single page files will be named - if you set **%d** docuPrinter will add the page number to the file name. You can also control the number of digits used in the file name by replacing %d with %0Nd where N is the number of digits you want to have, for example %03d will force docuPrinter to produce files with names like this: *'filename001.tif', ... , 'filename010.tif', ...* %04d will produce: *'filename0001.tif', ... , 'filename0010.tif', ...*

Syntax

DPSDK.FileMask = value

Example

DPSDK.FileMask = "%d"

Data type: **String**

Note: Will work with docuPrinter Pro and TSE.

User Interface specific properties

HideSaveAsWindow

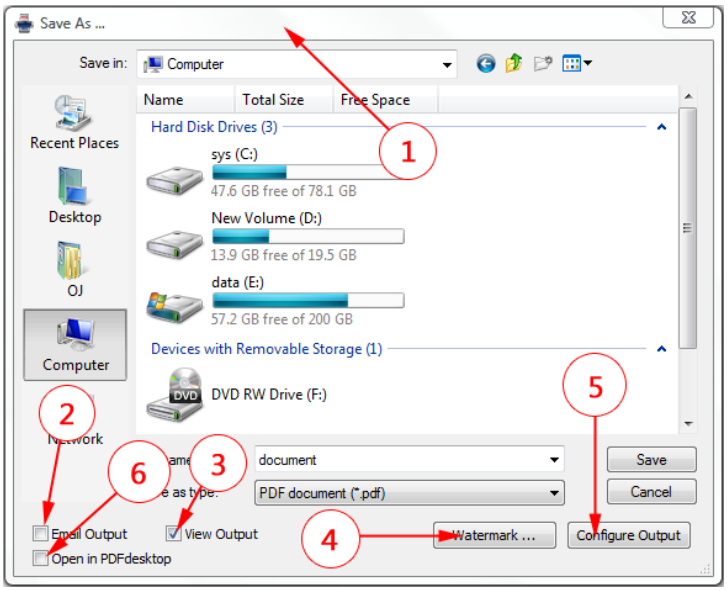
Specifies whether to show or hide the docuPrinter "Save As" window.

Syntax

DPSDK.HideSaveAsWindow = value

Data type: **Boolean**

Note: (1) on the screenshot.



HideWatermarkButton

Specifies whether to show or hide the "Watermark" button.

Syntax

DPSDK.HideWatermarkButton = value

Data type: **Boolean**

Note: (4) on the screenshot.

HideConfigureButton

Specifies whether to show or hide the "Configure" button.

Syntax

DPSDK.HideConfigureButton = value

Data type: **Boolean**

Note: (5) on the screenshot.

HideEmailCheckBox

Specifies whether to show or hide the "Email Output" checkbox. Note: (2) on the screenshot.

Syntax

DPSDK.HideEmailCheckBox = value

Data type: **Boolean**

HideViewCheckBox

Specifies whether to show or hide the "View Output" checkbox. Note: (3) on the screenshot.

Syntax

DPSDK.HideViewCheckBox = value

Data type: **Boolean**

HidePDFdesktopCheckBox

Specifies whether to show or hide the "Open in PDFdesktop" checkbox. Note: (6) on the screenshot.

Syntax

DPSDK.HidePDFdesktopCheckBox = value

Data type: **Boolean**

Note: Will not work with docuPrinter LT.

DefaultAction

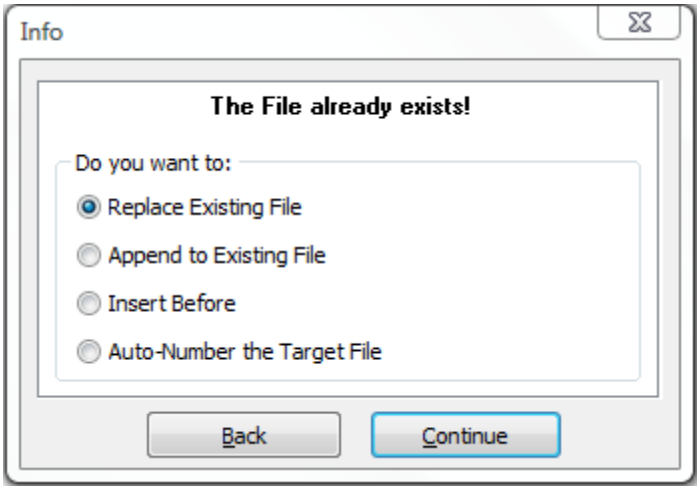
Specifies whether to show the
"File already exists" dialog.
Possible values:

0 - show dialog;
1 - hide dialog and replace existing file;
2 - hide dialog and append to existing file;
3 - hide dialog and insert before in the existing file;
4 - hide dialog and auto-number target file;

Syntax

DPSDK.DefaultAction = value

Data type: **Long**



ViewDocumentAfterConversion

Specifies whether docuPrinter should open the document after the conversion in the default system viewer.
Will have no effect if there is no viewer associated with this particular output format in windows.

Syntax

DPSDK.ViewDocumentAfterConversion = value

Data type: **Boolean**

EmailDocumentAfterConversion

Specifies whether docuPrinter should open the document after the conversion in the default system email client.
Will have no effect if there is no default email client defined in windows.

Syntax

DPSDK.EmailDocumentAfterConversion = value

Data type: **Boolean**

OpenInPDFdesktopAfterConversion

Specifies whether docuPrinter should open the document after the conversion in Neevia PDFdesktop.

Syntax

DPSDK.OpenInPDFdesktopAfterConversion = value

Data type: **Boolean**
Note: Will work with docuPrinter Pro and TSE.

Methods

NewGUID
Return a unique identifier.

Syntax

value= DPSDK.NewGUID

Data Type: String

GetDefaultPrinter
Return the default system printer.

Syntax

value = DPSDK.GetDefaultPrinter

SetDefaultPrinter
Sets the default system printer.

Syntax

Res = DPSDK.SetDefaultPrinter(printerName)

Remarks

Res<>0 on error.

BackupSettings
Backup docuPrinter settings.

Syntax

Res = DPSDK.BackupSettings

Remarks

Res<>0 on error.

RestoreSettings
Restores docuPrinter settings previously backed up using the BackupSettings method.

Syntax

Res = DPSDK.RestoreSettings

Remarks

Res<>0 on error.

ApplySettings
Applies the settings to the docuPrinter.

Syntax

Res = DPSDK.ApplySettings

Remarks

Res<>0 on error.

Create

Call this method to create the output document.

Syntax

Res = DPSDK.Create(timeOut)

Parameters

timeOut – Optional. Document creation timeout (in seconds) - if the output document is not created within the specified timeout period, Create will throw an error.

Remarks

Res<>0 on error.

TrueTypeFontDownloadOption

Sets the docuPrinter TrueType font download option.
Possible values:
0 – default
1 – download as outline
2 – download as bitmap
3 – download as native TrueType

Syntax

result = DPSDK.TrueTypeFontDownloadOption(option)

Note: MacOS X Preview doesn't support PDF files with custom true type font encoding - Acrobat Reader for MacOS X displays this type of PDF documents without any problem. A simple workaround is to force docuPrinter to produce PDF files with type1 (PostScript) fonts inside which are fully supported by MacOS X Preview - for this set TrueTypeFontDownloadOption=1.

FileDelete

Delete file from disk.

Syntax

result = DPSDK.FileDelete(filename)

FileInUse

Check if the file is locked by another application.

Syntax

result = DPSDK.FileInUse(filename)

FileExists

Check if the file exists.

Syntax

result = DPSDK.FileExists(filename)

FileCopy

Copy file to destination.

Syntax

result = DPSDK.FileCopy(inputfile, outputfile)

docuPrinter Word macro

Class ID

docuPrinter.WordMacro

Example:

Visual Basic: **Set DPWORD = CreateObject("docuPrinter.WordMacro")**

Delphi: **DPWORD := CreateOLEObject('docuPrinter.WordMacro');**

ASP: **Set DPWORD = Server.CreateObject("docuPrinter.WordMacro")**

VC#: first add a reference in your project to docuPrinter library

docuPrinter.WordMacro DPWORD = new docuPrinter.WordMacro();

Properties

CHBookmarks

Specifies whether docuPrinter should convert word headings into PDF bookmarks.
Possible values: **true, false**

Syntax

DPWORD.CHBookmarks = value

CInternetLink

Specifies whether docuPrinter should convert word hyperlinks into PDF links.
Possible values: **true, false**

Syntax

DPWORD.CInternetLink = value

CCrosRef

Specifies whether docuPrinter should convert MS Word cross-reference links into PDF links.
Possible values: **true, false**

Syntax

DPWORD.CCrosRef = value

CCrosDoc

Specifies whether docuPrinter should convert MS Word cross-document links into PDF links.
Possible values: **true, false**

Syntax

DPWORD.CCrosDoc = value

LinkFootEnd

Specifies whether docuPrinter should convert MS Word footnotes and endnotes into PDF links to the respective citations.
Possible values: **true, false**

Syntax

DPWORD.LinkFootEnd = value

CWordTextBox

Specifies whether docuPrinter should convert Word text boxes into PDF article threads.
Possible values: **true, false**

Syntax

DPWORD.CWordTextBox = value

CDocInfo

Specifies whether docuPrinter should convert MS Word document info (such as author and keywords) into PDF info.
Possible values: **true, false**

Syntax

DPWORD.CDocInfo = value

LinkType

Specifies how the PDF links should look.
Possible values:

0 - invisible links;
1 - visible links with thin border;
2 - visible links with thick border;

Syntax

DPWORD.LinkType = value

LinkHighlight

Specifies how the PDF links should be highlighted when they are clicked.
Possible values:

0 - None
1 - Invert
2 - Outline
3 - Inset

Syntax

DPWORD.LinkHighlight = value

LinkColor

Specifies the PDF link color.
Possible values:

0 – Black
1 – Red
2 – Green
3 - Yellow
4 – Blue
5 - Magenta
6 – Cyan
7 - White

Syntax

DPWORD.LinkColor = value

LinkStyle

Specifies the PDF link border style.
Possible values: 0 - Solid, 1 - Dashed

Syntax

DPWORD.LinkStyle = value

CBookNameDest

Specifies whether docuPrinter should convert MS Word bookmarks into PDF named destinations.
Possible values: **true, false**

Syntax

DPWORD.CBookNameDest = value

BookmarkDepth

Possible values: **0..9**

Syntax

DPWORD.BookmarkDepth = value

BookmarkMagn

Specifies the open magnification of the bookmark destination.
Possible values:

0 - Inherit Zoom
1 - Fit page width to window
2 - Fit page height to window
3 - Fit page to window

Syntax

DPWORD.BookmarkMagn = value

CFormFields
Specifies whether docuPrinter should convert MS Word form fields into PDF form fields.
Possible values: **true, false**

Syntax

```
DPWORD.CFormFields = value
```

RenameFormFields
Specifies whether docuPrinter should automatically rename form fields that have the same names so all the fields in the PDF document will have unique names.
Possible values: **true, false**

Syntax

```
DPWORD.RenameFormFields = value
```

HideTextInputBorder
Specifies whether docuPrinter should disable the text field border in the PDF file.
Possible values: **true, false**

Syntax

```
DPWORD.HideTextInputBorder = value
```

HideCheckBoxBorder
Specifies whether docuPrinter should disable the checkbox field border in the PDF file.
Possible values: **true, false**

Syntax

```
DPWORD.HideCheckBoxBorder = value
```

HideDropDownBorder
Specifies whether docuPrinter should disable the combobox/listbox field border in the PDF file.
Possible values: **true, false**

Syntax

```
DPWORD.HideDropDownBorder = value
```

CComNotes
Specifies whether docuPrinter should convert MS Word comments into PDF notes.
Possible values: **true, false**

Syntax

```
DPWORD.CComNotes = value
```

Methods

ConvertDocument
Call this method to convert the specified Word document.

Syntax

```
Res = DPWORD.ConvertDocument ( fileToConvert )
```

docuPrinter Excel macro

Class ID

docuPrinter.ExcelMacro

Example:

Visual Basic: **Set DPXLS = CreateObject("docuPrinter.ExcelMacro")**

Delphi: **DPXLS := CreateOLEObject('docuPrinter.ExcelMacro');**

VC#: first add a reference in your project to docuPrinter library

docuPrinter.ExcelMacro DPXLS = new docuPrinter.ExcelMacro();

Properties

CHBookmarks

Specifies whether docuPrinter should convert excel worksheet names into PDF bookmarks.
Possible values: **true, false**

Syntax

DPXLS.CHBookmarks = value

CInternetLink

Specifies whether docuPrinter should convert excel hyperlinks into PDF links.
Possible values: **true, false**

Syntax

DPXLS.CInternetLink = value

CDocInfo

Specifies whether docuPrinter should convert MS Excel document info (such as author and keywords) into PDF info.
Possible values: **true, false**

Syntax

DPXLS.CDocInfo = value

LinkType

Specifies how the PDF links should look.
Possible values:

- **0** - invisible links;
- **1** - visible links with thin border;
- **2** - visible links with thick border;

Syntax

DPXLS.LinkType = value

LinkHighlight

Specifies how the PDF links should be highlighted when they are clicked.
Possible values:

4 - None
5 - Invert
6 - Outline
7 - Inset

Syntax

DPXLS.LinkHighlight = value

LinkColor

Specifies the PDF link color.
Possible values:

0 - Black
1 - Red
2 - Green
3 - Yellow
4 - Blue
5 - Magenta
6 - Cyan
7 - White

Syntax

DPXLS.LinkColor = value

LinkStyle

Specifies the PDF link border style.
Possible values: 0 - Solid, 1 - Dashed

Syntax

DPXLS.LinkStyle = value

CBookNameDest

Specifies whether docuPrinter should convert MS Excel worksheet names into PDF named destinations.
Possible values: **true, false**

Syntax

```
DPXLS.CBookNameDest = value
```

BookmarkDepth

Possible values: **0..9**

Syntax

```
DPXLS.BookmarkDepth = value
```

BookmarkMagn

Specifies the open magnification of the bookmark destination.
Possible values:

- **4** - Inherit Zoom
- **5** – Fit page width to window
- **6** – Fit page height to window
- **7** – Fit page to window

Syntax

```
DPXLS.BookmarkMagn = value
```

Methods

ConvertDocument

Call this method to convert the specified Excel workbook.

Syntax

```
Res = DPXLS.ConvertDocument ( fileToConvert )
```

docuPrinter PowerPoint macro

Class ID

docuPrinter.PowerPointMacro

Example:

Visual Basic: **Set DP_PPT = CreateObject("docuPrinter.PowerPointMacro")**

Delphi: **DP_PPT := CreateOLEObject('docuPrinter.PowerPointMacro');**

VC#: first add a reference in your project to docuPrinter library

docuPrinter.PowerPointMacro DP_PPT = new docuPrinter.PowerPointMacro();

Properties

CTransition

Specifies whether docuPrinter should convert PowerPoint slide transition effects into PDF page effects.
Possible values: **true, false**

Syntax

DP_PPT.CTransition = value

CHBookmarks

Specifies whether docuPrinter should convert PowerPoint slide names into PDF bookmarks.
Possible values: **true, false**

Syntax

DP_PPT.CHBookmarks = value

CInternetLink

Specifies whether docuPrinter should convert PowerPoint hyperlinks into PDF links.
Possible values: **true, false**

Syntax

DP_PPT.CInternetLink = value

CDocInfo

Specifies whether docuPrinter should convert PowerPoint document info (such as author and keywords) into PDF info.
Possible values: **true, false**

Syntax

DPSDK.CDocInfo = value

LinkType

Specifies how the PDF link should look.
Possible values:

0 - invisible links;
1 - visible links with thin border;
2 - visible links with thick border;

Syntax

DP_PPT.LinkType = value

LinkHighlight

Specifies how the PDF links should be highlighted when they are clicked.
Possible values:

8 - None
9 - Invert
10 - Outline
11 - Inset

Syntax

DP_PPT.LinkHighlight = value

LinkColor

Specifies the PDF link color.
Possible values:

0 - Black
1 - Red
2 - Green
3 - Yellow
4 - Blue
5 - Magenta
6 - Cyan
7 - White

Syntax

DP_PPT.LinkColor = value

LinkStyle

Specifies the PDF link border style.
Possible values: 0 - Solid, 1 - Dashed

Syntax

DP_PPT.LinkStyle = value

BookmarkDepth

Possible values: **0..9**

Syntax

DP_PPT.BookmarkDepth = value

BookmarkMagn

Specifies the open magnification of the bookmark destination.
Possible values:

- **8** - Inherit Zoom
- **9** – Fit page width to window
- **10** – Fit page height to window
- **11** – Fit page to window

Syntax

DP_PPT.BookmarkMagn = value

BlackAndWhitePrinting

Specifies whether docuPrinter should convert the PowerPoint document into a B&W PDF file.
Possible values: **true, false**

Syntax

DP_PPT.BlackAndWhitePrinting = value

Methods

ConvertDocument

Call this method to convert the specified PowerPoint presentation.

Syntax

Res = DP_PPT.ConvertDocument (fileToConvert)

Examples

For more code samples visit http://neevia.com/support/examples/dpsdk/

Example 1. Create a PDF file from Visual Basic

```
Sub HelloWorld()

  Dim DPSDK : Set DPSDK = CreateObject("docuPrinter.SDK")

  DPSDK.BackupSettings

  Dim printerFound : printerFound = False

  Dim objPrinter
  For Each objPrinter In Printers
    If objPrinter.DeviceName = "docuPrinter" Then
      printerFound = True
      Set Printer = objPrinter
      Exit For
    End If
  Next

  If Not printerFound Then
    MsgBox "Printer not found!!!"
    Exit Sub
  End If

  DPSDK.DocumentOutputFormat = "PDF"
  DPSDK.DocumentOutputName = "demoVB"
  DPSDK.DocumentOutputFolder = "c:\"

  DPSDK.HideSaveAsWindow = true
  DPSDK.DefaultAction=1

  DPSDK.ApplySettings

  Printer.FontName = "Times New Roman"
  Printer.FontSize = 48
  Printer.Print "Hello from Visual Basic!!!"
  Printer.EndDoc

  RVal = DPSDK.Create ' Create output document
  If (RVal <> 0) Then MsgBox "Error while creating document!!!"

  DPSDK.RestoreSettings

  Set DPSDK = Nothing

  MsgBox "Done!!!"

End Sub
```

Example 2. Convert a MS Word document into PDF from Visual Basic

```
Sub WordConverter()

  Dim docToConvert : docToConvert="c:\test.doc"

  Dim DPSDK : Set DPSDK = CreateObject("docuPrinter.SDK")

  DPSDK.BackupSettings

  DPSDK.DocumentOutputFormat = "PDF"
  DPSDK.DocumentOutputName = "demoDOC"
  DPSDK.DocumentOutputFolder = "c:\"

  DPSDK.HideSaveAsWindow = true
  DPSDK.DefaultAction=1

  DPSDK.ApplySettings

  Dim MSWord : Set MSWord = CreateObject("Word.Application")
  MSWord.DisplayAlerts = False
  On Error Resume Next

  Dim NewDoc
  Set NewDoc = MSWord.Documents.Open(docToConvert, False, True)
  If Err<>0 Then
    Set MSWord = Nothing
    Exit Sub
  End If

  Dim MSWordDialog : Set MSWordDialog = MSWord.Dialogs(97)
  MSWordDialog.Printer = "docuPrinter"
  MSWordDialog.DoNotSetAsSysDefault = 1
  MSWordDialog.Execute

  NewDoc.PrintOut False

  NewDoc.Close False
  MSWord.Quit False
  Set MSWord = Nothing

  Dim RVal : RVal = DPSDK.Create ' Create output document
  If (RVal <> 0) Then MsgBox "Error while creating the document!!!"

  DPSDK.RestoreSettings

  Set DPSDK = Nothing

  MsgBox "Done converting!!!"

End Sub
```

Example 3. Convert a MS Word document into PDF from Visual Basic (using the docuPrinter MS Word Macro)

```
Sub WordConverter()

  Dim docToConvert : docToConvert="c:\test.doc"

   Dim DPSDK : Set DPSDK = CreateObject("docuPrinter.SDK")

   DPSDK.BackupSettings

   DPSDK.DocumentOutputFormat = "PDF"
   DPSDK.DocumentOutputName = "demoDOC"
   DPSDK.DocumentOutputFolder = "c:\"

   DPSDK.HideSaveAsWindow = true
   DPSDK.DefaultAction=1

   DPSDK.ApplySettings

   Dim DPWORD : Set DPWORD = CreateObject("docuPrinter.WordMacro")
   DPWORD.CHBookmarks=true
   DPWORD.CInternetLink=true
   DPWORD.CCrosRef=true
   DPWORD.CCrosDoc=true
   DPWORD.LinkFootEnd=true
   DPWORD.CWordTextBox=true
   DPWORD.CDocInfo=true
   DPWORD.CBookNameDest=true
   DPWORD.CComNotes=true
   DPWORD.ConvertDocument docToConvert

   Set DPWORD = Nothing

   DPSDK.RestoreSettings

   Set DPSDK = Nothing

   MsgBox "Done converting!!!"

End Sub
```

Example 4. Convert a MS Excel document into PDF from Visual Basic

```
Sub ExcelConverter()

  Dim docToConvert : docToConvert="c:\test.xls"

  Dim DPSDK : Set  DPSDK = CreateObject("docuPrinter.SDK")

  DPSDK.BackupSettings

  DPSDK.DocumentOutputFormat = "PDF"
  DPSDK.DocumentOutputName = "demoXLS"
  DPSDK.DocumentOutputFolder = "c:\"

  DPSDK.HideSaveAsWindow = true
  DPSDK.DefaultAction=1

  DPSDK.ApplySettings

  Dim MSExcel : Set MSExcel = CreateObject("Excel.Application")
  MSExcel.DisplayAlerts = False

  Dim XLDoc : Set XLDoc = MSExcel.Workbooks.Open(docToConvert, 0, True)

  XLDoc.Activate
  XLDoc.PrintOut ,,, False, "docuPrinter"
  XLDoc.Saved = True
  XLDoc.Close
  MSExcel.Quit
  Set MSExcel = Nothing

  Dim RVal : RVal = DPSDK.Create ' Create output document
  If (RVal <> 0) Then MsgBox "Error while creating the document!!!"

  DPSDK.RestoreSettings

  Set DPSDK = Nothing

  MsgBox "Done converting!!!"

End Sub
```

Example 5. Convert a MS PowerPoint document into PDF from Visual Basic

```
Sub PowerPointConverter()

  Dim documentToConvert : documentToConvert="c:\test.ppt"

  Dim DPSDK : Set  DPSDK = CreateObject("docuPrinter.SDK")

  DPSDK.BackupSettings

  DPSDK.DocumentOutputFormat = "PDF"
  DPSDK.DocumentOutputName = "demoPPT"
  DPSDK.DocumentOutputFolder = "c:\"

  DPSDK.PDFAutoRotatePage = "PageByPage"

  DPSDK.HideSaveAsWindow = true
  DPSDK.DefaultAction=1

  DPSDK.ApplySettings

  Dim MSPowerPoint
  Set MSPowerPoint = CreateObject("PowerPoint.Application")

  Dim PPTDoc
  Set PPTDoc = MSPowerPoint.Presentations.Open(documentToConvert, -1, 0, 0)
  PPTDoc.PrintOptions.PrintInBackground=0
  PPTDoc.PrintOptions.PrintColorType=1
  PPTDoc.PrintOptions.ActivePrinter="docuPrinter"
  PPTDoc.PrintOut 0
  PPTDoc.Close
  MSPowerPoint.Quit
  Set MSPowerPoint = Nothing

  Dim RVal : RVal = DPSDK.Create ' Create output document
  If (RVal <> 0) Then MsgBox "Error while creating the document!!!"

  DPSDK.RestoreSettings

  Set DPSDK = Nothing

  MsgBox "Done converting!!!"

End Sub
```

Example 6. Convert a PowerPoint document into PDF from Visual Basic
(using the docuPrinter PowerPoint Macro)

```
Sub PowerPointConverter()

  Dim documentToConvert : documentToConvert="c:\test.ppt"

  Dim DPSDK : Set  DPSDK = CreateObject("docuPrinter.SDK")

  DPSDK.BackupSettings

  DPSDK.DocumentOutputFormat = "PDF"
  DPSDK.DocumentOutputName = "demoPPT2"
  DPSDK.DocumentOutputFolder = "c:\"

  DPSDK.PDFAutoRotatePage = "PageByPage"

  DPSDK.HideSaveAsWindow = true
  DPSDK.DefaultAction=1

  DPSDK.ApplySettings

  Dim DPPPT : Set DPPPT = CreateObject("docuPrinter.PowerPointMacro")

  DPPPT.CTransition=true
  DPPPT.CHBookmarks=true
  DPPPT.CInternetLink=true
  DPPPT.CDocInfo=true
  DPPPT.ConvertDocument documentToConvert

  Set DPPPT = Nothing

  DPSDK.RestoreSettings

  Set DPSDK = Nothing

  MsgBox "Done converting!!!"

End Sub
```

Example 7. Convert a MS Access report into PDF from Visual Basic

```
Sub AccessConverter()

  Dim DPSDK : Set DPSDK = CreateObject("docuPrinter.SDK")

  DPSDK.BackupSettings

  DPSDK.DocumentOutputFormat = "PS"
  DPSDK.DocumentOutputName = "demoAccess"
  DPSDK.DocumentOutputFolder = "c:\"

  DPSDK.DocumentResolution = 300

  DPSDK.HideSaveAsWindow = true
  DPSDK.DefaultAction=1

  DPSDK.ApplySettings

  Dim objAccess
  Set objAccess = CreateObject("Access.Application")

  Dim defPrinter : defPrinter=DPSDK.GetDefaultPrinter

  DPSDK.SetDefaultPrinter "docuPrinter"
  objAccess.OpenCurrentDatabase "c:\access.mdb", true

  objAccess.DoCmd.OpenReport "rptCatalog", 0
  'rptCtatalog is the repport name

  objAccess.Quit 2
  Set objAccess=nothing

  RVal = DPSDK.Create ' Create output document
  If (RVal <> 0) Then MsgBox "Error. Create returns "+CStr(Rval)

  DPSDK.SetDefaultPrinter defPrinter

  DPSDK.RestoreSettings

  Set DPSDK = Nothing

  MsgBox "Done converting!!!"

End Sub
```

Example 8. Convert an URL / HTML into PDF from Visual Basic

```
Sub URLConverter()

   Dim IE : Set IE = CreateObject("InternetExplorer.Application")
   IE.Visible=true

   Dim DPSDK : Set DPSDK = CreateObject("docuPrinter.SDK")

   DPSDK.BackupSettings

   IE.navigate2 "http:\\www.neevia.com"
   DPSDK.doSleep 100

   While (IE.ReadyState<>4) or (IE.Busy)
     DPSDK.doSleep 100
   Wend

   DPSDK.DocumentOutputFormat="PDF"
   DPSDK.DocumentOutputFolder="c:\"
   DPSDK.DocumentOutputName="testURL"

   DPSDK.HideSaveAsWindow = true
   DPSDK.DefaultAction=1

   DPSDK.ApplySettings

   Dim defPrinter : defPrinter = DPSDK.GetDefaultPrinter
   DPSDK.SetDefaultPrinter "docuPrinter"

   IE.ExecWB 6,2

   Dim RVal : RVal = DPSDK.Create
   If (RVal <> 0) Then MsgBox "Error while creating the document!!!"

   IE.Quit

   DPSDK.SetDefaultPrinter defPrinter

   Set IE=Nothing

   DPSDK.RestoreSettings

   Set DPSDK=Nothing

   MsgBox "Done converting!!!"

End Sub
```